International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 4 | Issue 2 | ISSN : 2456-3307

An Optimal Strategy for Evaluating Continuous Top-k Monitoring on Document Streams

V. Hema Priya [*1], Ponduri Siva Parvathi [2]
[1*] hemapriyamtech@gmail.com,siva.ponduri1609@gmail.com[2]

ABSTRACT

The proficient preparing of report streams assumes a vital part in numerous data separating frameworks. Developing applications, for example, news refresh sifting and informal community warnings, request to give end-clients the most significant substance to their inclinations. In this work, client inclinations are demonstrated by an arrangement of watchwords. A focal server screens the record stream and consistently reports to every client the best k archives that are most pertinent to her catchphrases. Our goal is to help substantial quantities of clients and high stream rates while invigorating the best k comes about quickly. Our answer forsakes the conventional recurrence requested ordering approach. Rather, it takes after an identifier-requesting worldview that suits better the idea of the issue. At the point when supplemented with a novel, locally versatile strategy, our technique offers (I) demonstrated optimality w.r.t. the quantity of considered questions per stream occasion, and (ii) a request for extent shorter reaction time (i.e., time to invigorate the inquiry comes about) than the present best in class.

Keywords: *Top-k query, continuous query, document stream*

I. INTRODUCTION

N the era of big data, the amount of information made available to users far exceeds their capacity to discover and understand it. For instance, a user on Twitter may receive an overwhelming volume of notifications if her mes- sage is retweeted by too many people in a short period. Moreover, the timeliness of information filtering and deliv- ery is of great importance. For example, a user would like to receive instant updates of the hottest topics on social news and entertainment websites (e.g., on reddit.com). Thus, the efficient filtering and monitoring of rapid streams is key tomany emerging applications.

We consider continuous top-k queries on documents (CTQDs), a topic which has received a lot of attention recently [1], [2], [3]. In this context, a central server monitors a document stream and hosts CTQDs from various users. Each CTQD specifies a set of keywords, as explicitly given by the issuing user or extracted from Their online behaviour [4], [5]. The task of the server is to continuously refresh for snapshot top-k queries revealed that, for sparse types of data, it may be more effec- tive to sort the lists of the inverted file by document ID [10], thus enabling “jumps” within the relevant lists, i.e., disre-garding contiguous fractions of the lists. This is an interest-ing fact, which however is not directly applicable to continuous top-k queries. An application of ID-ordering to document streams would incur costly index maintenance, and also it would require repetitive query reevaluation, as it entails no mechanism to reuse past query results in response to updates.

We propose an ID-ordering methodology for CTQDs. Our methodology involves three dimensions. First, we reverse the role of the

documents and the queries. That is, we index the (relatively static) queries and probe the streaming docu- ments against that index, in order to eliminate the need for index maintenance due to stream events. The general idea of indexing the queries instead of the data in a streaming con- text is commonly referred to as query indexing, and has been used for many types of continuous queries (e.g., [11]). Second, since we index user queries which, unlike the documents, typically comprise just a few terms (i.e., they are hugely sparse), we may effectively apply ID-ordering to the query index. The adaptation of ID-ordering to a query index, how- ever, is far from trivial and requires a careful redesign of its inner workings, as we explain in Section 4.2. By incorporat- ing the first two dimensions, we already have a preliminary CTQD method (albeit just a stepping stone to our complete, most comprehensive solution), termed Reverse ID-Ordering (RIO). RIO is already faster than existing CTQD approaches, but we do not stop there. Third, we complement RIO with a novel, locally adaptive technique that produces tighter proc-essing bounds. This technique renders the overall CTQD method optimal w.r.t. the number of considered queries per stream event, i.e., we prove that it computes the score of an arriving document w.r.t. the smallest possible number of queries, for any algorithm that follows the ID-ordering para- digm and guarantees correctness. The resulting method is our most advanced technique, called Minimal RIO (MRIO).

Through an extensive experimental evaluation with streams of real documents, we demonstrate that MRIO out- performs the current state-of-the-art CTQD solution by an order of magnitude. Furthermore, the "internal" comparison between MRIO and RIO reveals that the vast performance improvements achieved are primarily due to the third dimension sketched above, i.e., due to our locally adaptive technique. The contributions we make in this paper are summarized as follows:

- Our advanced approach (MRIO) outperforms the current state-of-the-art by one order of magnitude.
- MRIO employs novel bounds that offer proven optimality w.r.t. the number of considered queries per stream event.
- MRIO is more than two times faster than RIO, dem- onstrating that a skillful adaptation of ID-ordering to CTQDs alone (as in RIO) is not enough to derive the improvements achieved in this work.
- We further improve the performance of MRIO by restructuring its query index (i.e., rearranging the queries inside) to better exploit locality and strengthen the pruning effectiveness of its bounds.

Our evaluation has a broader experimental value too, because it involves (besides the state-of-the-art for CTQDs) methods for different formulations, which perform competitively, and were never put in the same testbed before.

II. RELATED WORK

In information filtering the objective is to remove from an information stream those items that are of no inter-est to the end users. Information filtering approaches have been studied for text streams [12], however, their focus is to determine an appropriate relevance threshold, based on the user's profile and the stream's character-istics [13]. The actual filtering involves fixed thresholds (and therefore binary relevance assessments per stream item), rather than relative similarity and ranking.

Publish subscribe is a messaging pattern where the publishers of messages categorize their messages into

classes, and the subscribers receive only those messages that fall in their classes of interest [14], [15]. Unlike CTQD, there is typically a set of predefined classes (instead of terms) and there is no notion of relative rank-ing. [16] does consider relative similarity, however, its goal is to identify the k most relevant queries for every newly published message. [17] proposes a probabilistic algorithm that keeps a select subset of the messages in a sliding window to support approximate top-k process-ing. Still in the publish subscribe setting, [18] considers the social annotation of news articles. Specifically, given a set of news stories (documents), it maintains for each of them the k most related tweets posted. Although in the documents (news stories) play the role of the standing queries, it could be applied to our setting (by treating user queries as news stories), although it is not tailored to it. We include this method in our experiments, abbreviated as TPS (for top-k publish subscribe).

The top-k query is relevant to our work. Given a set of options and a scoring function defined over their attributes, the goal is to report the k options with the highest scores. Top-k processing methods have been extensively studied in relational databases; [19] offers an extensive survey. Among them, the threshold algorithm [9] is central to our competitors. It assumes that the options are indexed by a number of lists, each of which is respon-sible for one option attribute, and keeps options sorted in descending order of that attribute. The main idea is to consider options from the sorted lists in a round-robin fashion and maintain an upper bound (threshold) for the score of any unseen option. The algorithm terminates when the k-th best option found so far scores no lower than the threshold.

In the context of text search engines, similarity search is typically framed as a top-k problem over a set of docu-ments. Terms (in queries and documents) are treated as attributes, weighted based on a standard scheme (e.g., tf-idf or Okapi BM25). The score of a document for a query is defined as a function over their common terms, such as cosine similarity. To facilitate search, the documents are indexed by an inverted file; [8] surveys different types of inverted files and query processing techniques. The inverted file includes a sorted list per term. In the frequency-ordering paradigm, the sorting key is term frequency (weight), whereas in ID-ordering it is the document ID. In the former case, processing follows similar principles to the threshold algorithm in order to consider only the top parts of the sorted lists. In the latter case, the lists are read in their entirety but jumps over ID ranges are made possible; in Section 4.1, we describe in more detail the most efficient processing approach in this paradigm [10], [20].

Continuous versions of the top-k query have also been studied. Top-k monitoring was originally addressed over a stream of low-dimensional records [21], [22]. The pro-posed methods relied on spatial indices and geometric reasoning (e.g., dual space transformations), and were thus tailored to data in just a handful of dimensions. Bound by the dimensionality curse, these approaches are not applicable to document streams, because if terms were dealt with as attributes, dimensionality would be in the order of hundreds of thousands.

Rao et al. [23] consider streams of documents, but address a special version of continuous top-k queries where the query weights are equal (equivalently, the query terms are unweighted). In this version of the problem, if the search terms in a query $\mathbf{q}$ are a superset of those in another $\mathbf{q}^0$, then the score of a document w.r.t. $\mathbf{q}$ is always larger than its score w.r.t. $\mathbf{q}^0$. This means that if we compute the score of

a stream document **d** w.r.t. **q**, and that score is already smaller than the score of the k-th document in the result of $\mathbf{q}^0$, we can directly infer that $\mathbf{q}^0$ is not affected by **d**. The proposed solution utilizes this "coverage" relationship between queries to safely ignore some of them when a document streams in. It is inapplicable to our problem, where query weights are generally not equal. Even if an extension were possible, the chances of an ad-hoc user query being completely covered by another would be too slim.

Closest related to our work are methods for continuous top-k queries (with ad-hoc term weights) on document streams. [2] assumes the sliding window model and indexes the valid documents by a (frequency-ordered) inverted file. It uses the threshold algorithm to compute the initial top-k results, and maintains pointers in the sorted lists so as to resume processing from these positions when result refill is necessary. [1] proposes an approach that also relies on frequency-ordering and the threshold algorithm, but indexes the queries instead of the stream documents. It is shown to outperform [2] and is the current state-of-the-art. We refer to it as reverse threshold algorithm (RTA). The same authors extended RTA to heterogeneous scoring functions, by considering hotness in addition to similarity score [24].

Vouzoukidou et al. [3] propose a CTQD method, called SortQuer. For every term in the dictionary, they repre-sent each query **q** that includes the term as a point in a two-dimensional space – one axis corresponds to the score of the current top result document and the other axis to the query weight for that term. When a document (which includes the term) arrives, it is mapped to a region. Only queries that fall in that region could be affected by the document. Vouzoukidou et al. [3] eval-uate the k = 1 case; in that case, SortQuer outperforms RTA. Although SortQuer was designed with the k = 1 case in mind, it applies easily to k > 1 as well, thus we include it in our experiments, and offer a comprehensive evaluation against our methods and previous art.

Some stream processing frameworks for multidimensional objects are also related to our work. Koudas et al. [25] propose an approximate k-nearest neighbor monitoring technique for streams of low-dimensional points. However, their solution is inapplicable to CTQDs, because even if stream documents were mapped to points in term space, their dimensionality would be in the order of many thousands. Zhang et al. [26] study a shared processing framework for multiple aggregation queries on a stream. It is an interesting idea to share computations among queries. However, this work is in-applicable to CTQDs because it cannot handle weighted sum aggregates for arbitrary weights. That is, even if two CTQDs share some common terms, their respective weights for these terms are generally different.

III. EXSTING SYSTEM

In this section, we first define the similarity metric between queries and documents, and present the model of focusing on the fresher stream content. Next, we formalize the contin- uous top-k query on documents.

Similarity Measure

We treat the query (i.e., the set of keywords it specifies) and the documents as vectors. Letting T be the dictionary of all terms, a query or a document vector includes one weight per term in the dictionary.

That is, a query q and a docu- ment d are represented as

q ¼ hw1; w2; ... ; wjTji; d ¼ hf1; f2 ... ; fjTji:

The term weights wi and fi can be assigned to queries and documents, respectively, using any standard weighting scheme, such as tf-idf or Okapi BM25. Without loss of gen- erality, we assume that all query and document vectors are normalized to unit magnitude. Note that user queries typically include just a few terms, which means that the query vectors are extremely sparse (i.e., include numerous zero values). The documents, too, include just a fraction of the terms in the dictionary, thus, there are many zeros in their vectors.

As per common practice [27], the textual similarity between a query q and a document d is defined as the cosine similarity of their vectors

The cosine similarity measure cðq; dÞ takes values in ½0; 1];

the higher its value, the greater the textual similarity.

Document Freshness

In stream monitoring applications, the freshness of informa- tion is essential. Hence, a focusing on the most recent stream documents is required. The two prevalent formulations to achieve this focusing are the sliding window and the decay model. A sliding window only considers as valid the docu- ments that arrived most recently; the sliding window includes either a fixed number of documents (count-based version) or those that arrived within a fixed number of time units before current time (time-based). On the other hand, in the decay model the score of the documents drops over time by applying a decay function, so that the more recent documents are favored in query answering. Our work applies to both models, but the latter (and in particular its forward-decay version) is better suited to our targeted applications, as suggested in [3], [28]. The forward-decay model works as follows.

Consider a document d with arrival time td and cosine similarity cðq; dÞ to query q. The score of d w.r.t. q is defined as

Sðq; dÞ¼ cðq; dÞ=e—ZDtd;

where Z is an application-specified decay parameter, and Dtd is the difference between the document's arrival time td and a reference time t0 in the past (e.g., the system startup time). Unlike the vanilla decay model, this forward-decay formula- tion associates an invariable score to each arriving document (that is, Sðq; dÞ does not change over time)

while at the same time it effectively captures the freshness requirement of streaming applications [28]. Note that in this model, arriving documents receive increasingly larger scores as time goes by.1

Problem Definition

A stream of documents flows into a central processing server, which hosts a set of CTQDs. Each CTQD specifies a set of keywords (modeled as a query vector q) and a posi- tive integer k. For the sake of notation, we denote by m the number of keywords it specifies. The result of a CTQD includes the k stream documents with the highest scores Sðq; dÞ seen so far. The task of the stream server is to update all query results as new documents arrive. Document arrivals are referred to as stream events. The primary performance metric in our work is the

time required to refresh (update) all CTQD results in response to stream events.

Although in our default setting document arrivals are the only type of stream events, in Section 6 we consider the han- dling of other events types, such as query insertions, query deletions, and document expirations. The handling of docu- ment expirations enables, among others, the application of our framework to the sliding window model too.

IV. ID-ORDERING TECHNIQUES

Index & Query Processing for Snapshot Queries In this section, we overview the ID-ordering paradigm for snapshot queries. The documents are indexed by an inverted file, comprising a list Li for every term ti in the dictionary. Li holds an entry hdID; fii for every document that includes term ti (where dID is the ID of the document, and fi its weight for term ti). All lists are sorted in ascending document ID. The execution strategy to process a (snapshot) query q on this index evaluates the ocuments one after another from the sorted lists, but it performs "jumps" over zones of document IDs. The most efficient processing approach is Weighted AND [10], [20]; in the following we refer to this approach.

The maximum fi value in each list is pre-computed and stored with it – we denote it as mi. Posed a query, the algo- rithm executes in a number of iterations involving only the relevant lists. For every list Li, a cursor ci is used to store the ID of the next unconsidered document in the list. Assume

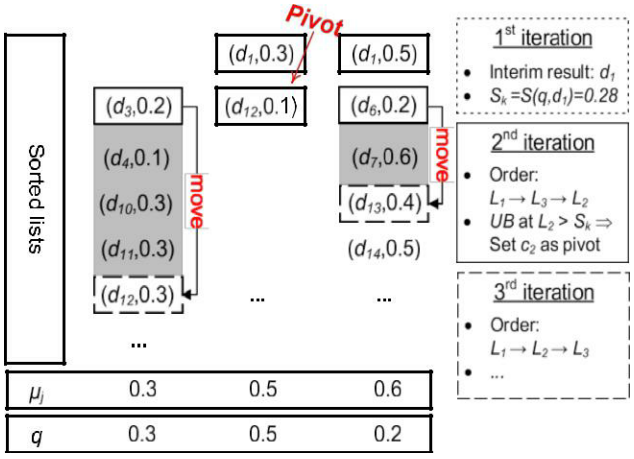

Fig. 1. Query processing in the ID-ordering paradigm.

That the query involves terms t1; t2; tm. At the beginning of an iteration, the processing order among the lists is decided based on their ci, i.e., by placing first the list whose cursor points at the smallest document ID, then the list whose cur- sor points at the next smallest document ID, etc. Assume that the processing order is L1 ! L2 ! .. . ! Lm (equivalently, in the beginning of the iteration c1 ≤ c2 ≤ ... ≤ cm). The invariant of the method is that, for every i 2 ½1; mÞ, any list after the ith in the processing order includes no entry for doc- ument IDs in ½c1; ciÞ.

IV. CONCLUSION

In this paper, we propose an adaptable system for the preparing of nonstop best k questions on report streams. A CTQD persistently reports the k most pertinent records to an arrangement of watchwords. CTQDs discover application in numerous developing applications, for example, email and news separating our preparatory approach, RIO, adjusts the ID-requesting worldview to the CTQD setting. An examination on RIO uncovers that the key factor that decides its execution is the quantity of emphases it executes. This inspires our propelled approach, MRIO, which decreases the quantity of cycles as well

as is demonstrated to limit it. We accomplish this by presenting the novel, locally versatile limits. Broad analyses with floods of genuine records exhibit that MRIO is a request of size speedier than the past cutting edge. A promising bearing for future work is to stretch out our approach to estimated top-k inquiries.

V. REFERENCES

1. P. Haghani, S. Michel, and K. Aberer, "The gist of everything new: Personalized top-k processing over web 2.0 streams," in Proc. 19th ACM Int. Conf. Inf. Knowl. Manage., 2010, pp. 489–498.
2. K. Mouratidis and H. Pang, "Efficient evaluation of continuous text search queries," IEEE Trans. Knowl. Data Eng., vol. 23, no. 10, pp. 1469–1482, Oct. 2011.
3. N. Vouzoukidou, B. Amann, and V. Christophides,"Processing continuous text queries featuring non-homogeneous scoring functions," in Proc. 21st ACM Int.Conf. Inf. Knowl. Manage., 2012, pp. 1065–1074.
4. A. Hoppe, "Automatic ontology-based user profile learning from heterogeneous web resources in a big data context,"Proc. VLDB Endowment, vol. 6, pp. 1428–1433, 2013.
5. A. Lacerda and N. Ziviani, "Building user profiles to improve user experience in recommender systems," in Proc. 6th ACM Int. Conf. Web Search Data Mining, 2013, pp. 759–764.
6. M. Busch, K. Gade, B. Larson, P. Lok, S. Luckenbill, and J.J. Lin, "Earlybird: Real-time search at twitter," inProc. IEEE 28th Int. Conf. Data Eng., 2012, pp. 1360–1369.
7. L. Wu, W. Lin, X. Xiao, and Y. Xu, "LSII: an indexing structure for exact real-time search on microblogs," in Proc.IEEE 29th Int. Conf. Data Eng., 2013, pp. 482–493.
8. J. Zobel and A. Moffat, "Inverted files for text search engines,"ACM Comput. Surv., vol. 38, no. 2, 2006, Art. no. 6.
9. R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algo- rithms for middleware," J. Comput. Syst. Sci., vol. 66, no. 4, pp. 614–656, 2003.

AUTHOR DETAILS

V.HEMA PRIYA is working an assistant professor in VIGNAN'S LARA INSTITUTE OF TECHNOLOGY&SCIENCE.Vadlamudi-522213 Guntur Dist.She has Experience in the teaching field For 2 years and her interested in research area data mining.

PODURI SIVA PARVATHI she is Currently pursuing MCA in MCA Department,Vignan's Lara Institute Of Technology&Science,Vadlamudi, Guntur, Andhra Pradesh, India. she received his Bachelor of science from ANU